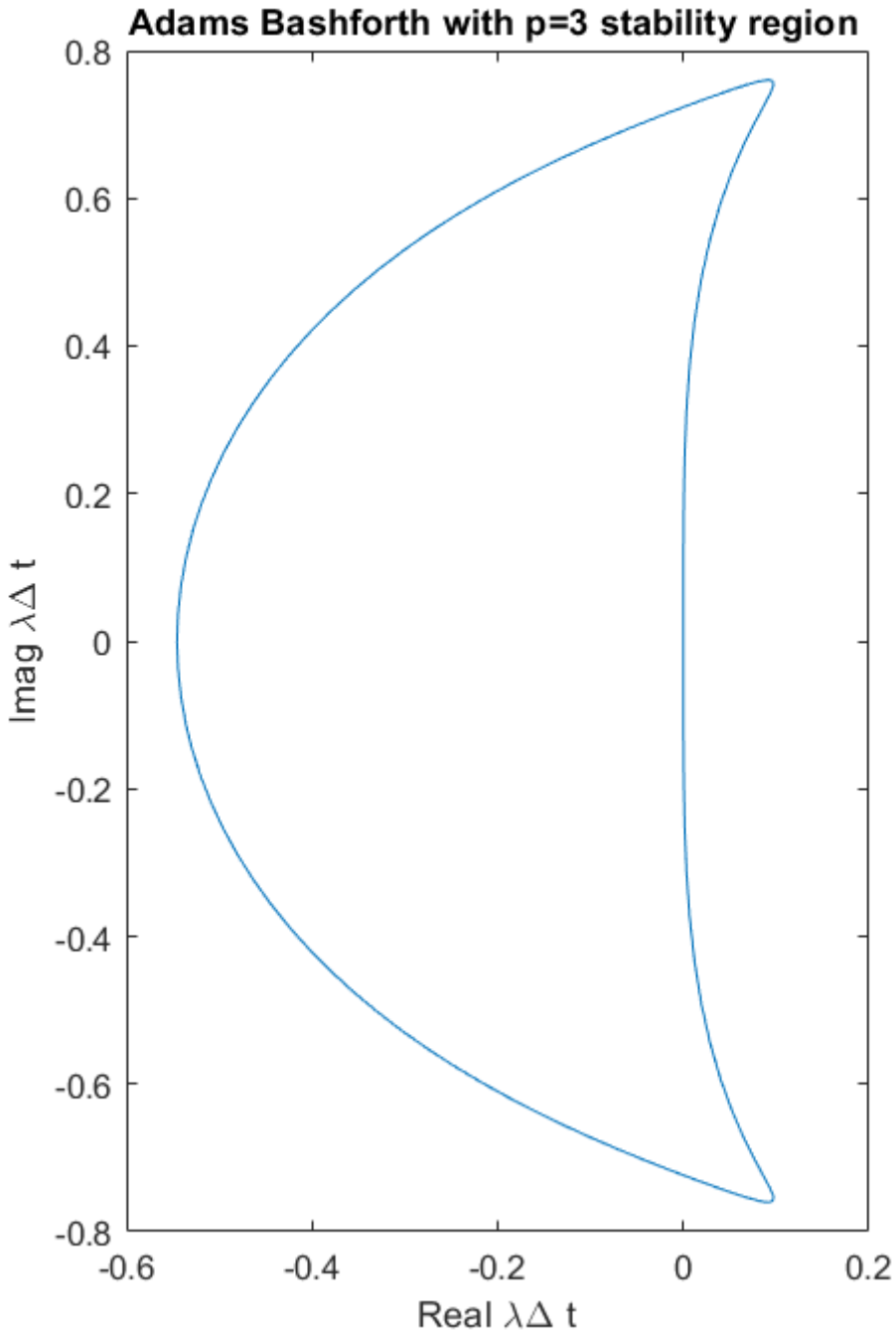
# Annex: Report for 16.90 PSet 2

*By Codrin Oneci*

1A. Use this code:

```
th = 0:pi/50:2*pi;
g=exp(1j*th);
z=zeros(1,length(g));
for i=1:length(g)
   c=g(i);
   z(i)=(12*c^3-12*c^2)/(23*c^2-16*c+5);
end
disp(g);
xunit = real(z);
yunit = imag(z);
h = plot(xunit, yunit);
title('Adams Bashforth with p=3 stability region ');
xlabel('Real \lambda\Delta t');
ylabel('Imag \lambda\Delta t');
```

The result is shown below:



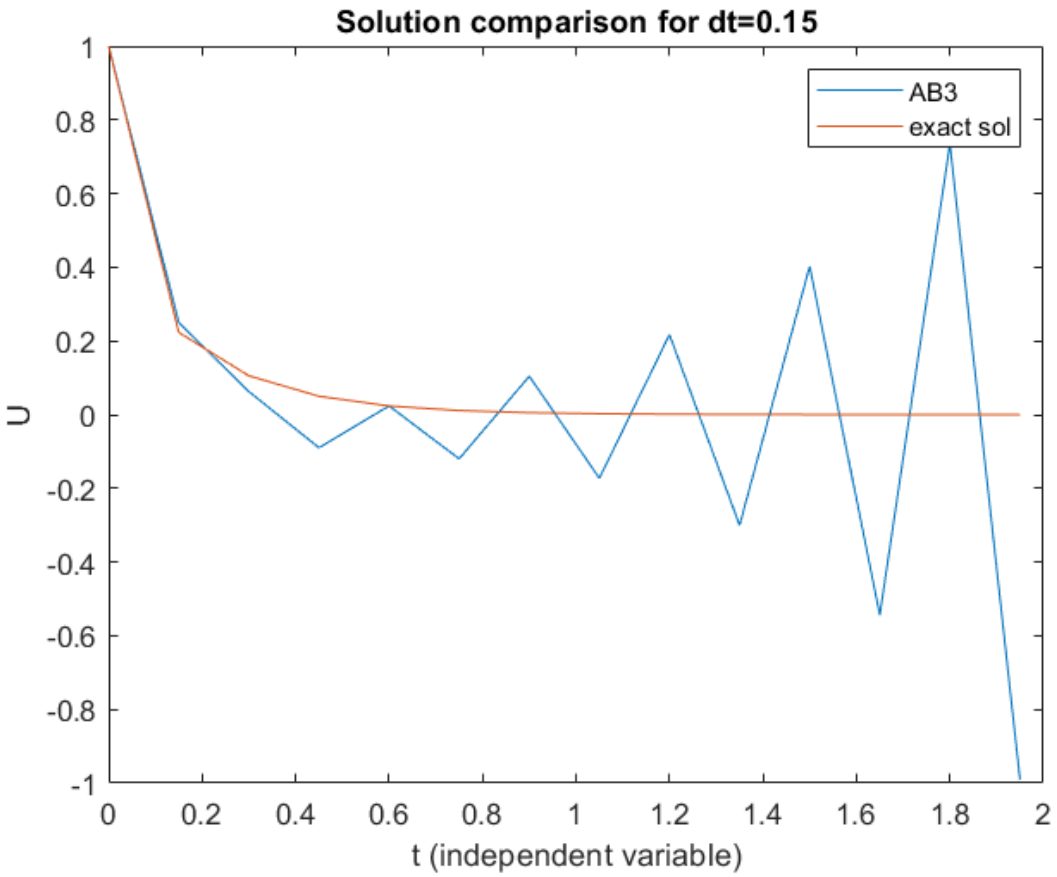

1B. Use this code:

```
u0=1;
dt=0.01; %timestep
Tfinal=2; %finish time
t=[0:dt:Tfinal]; %time vector
U_AB=zeros(1,length(t));U_AB(1)=u0;
U_exact=zeros(1,length(t));U_exact(1)=u0;
for i=2:4
   U_AB(i)=U_AB(i-1)+dt*odefun(U_AB(i-1));
```

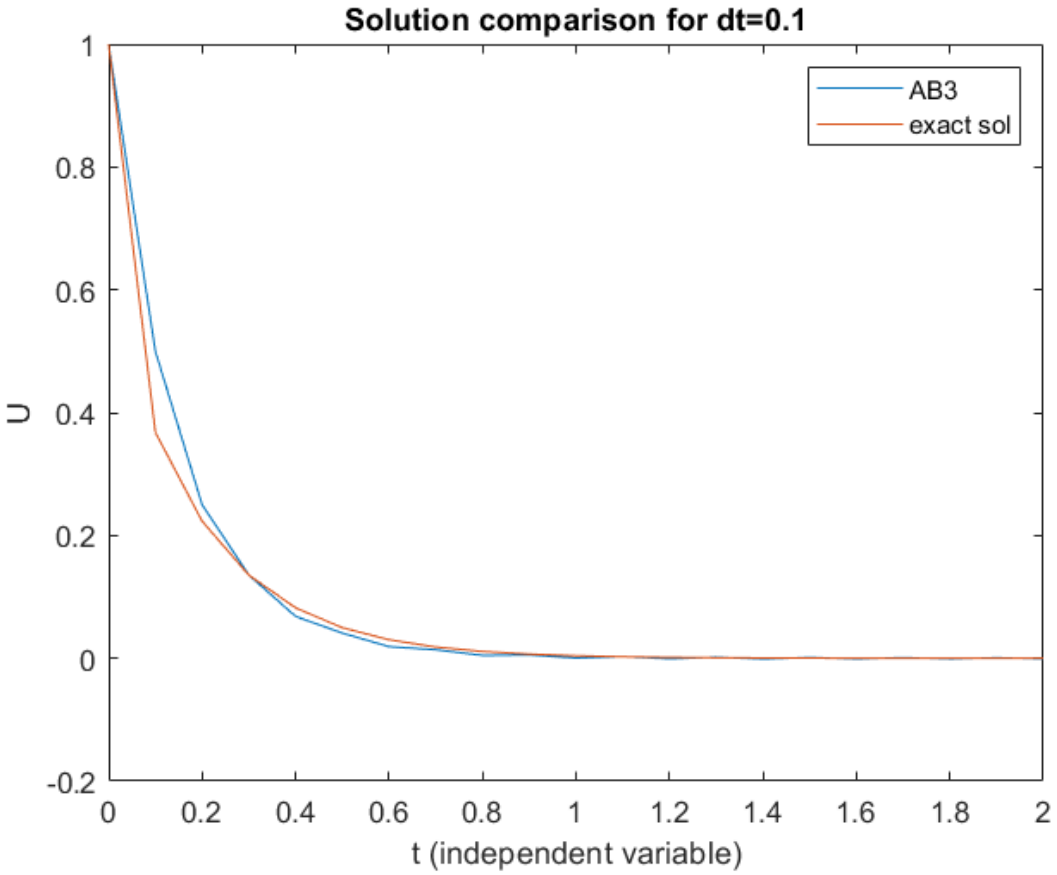
```
    U_exact(i)=exp(1)^(-5*i*dt);
end
for i=4:length(t)
    U_AB(i)=U_AB(i-1)+(dt/12)*(23*odefun(U_AB(i-1))-16*odefun(U_AB(i-2)) +5* odefun(U_AB(i-3)));
    U_exact(i)=exp(1)^(-5*i*dt);
end
figure(1);
plot(t,U_AB,'-',t,U_exact,'-')
legend('AB3','exact sol')
title('Solution comparison for dt=0.01');
xlabel('t (independent variable)');
ylabel('U');

function dudt=odefun(u)
dudt=-5*u
end
```

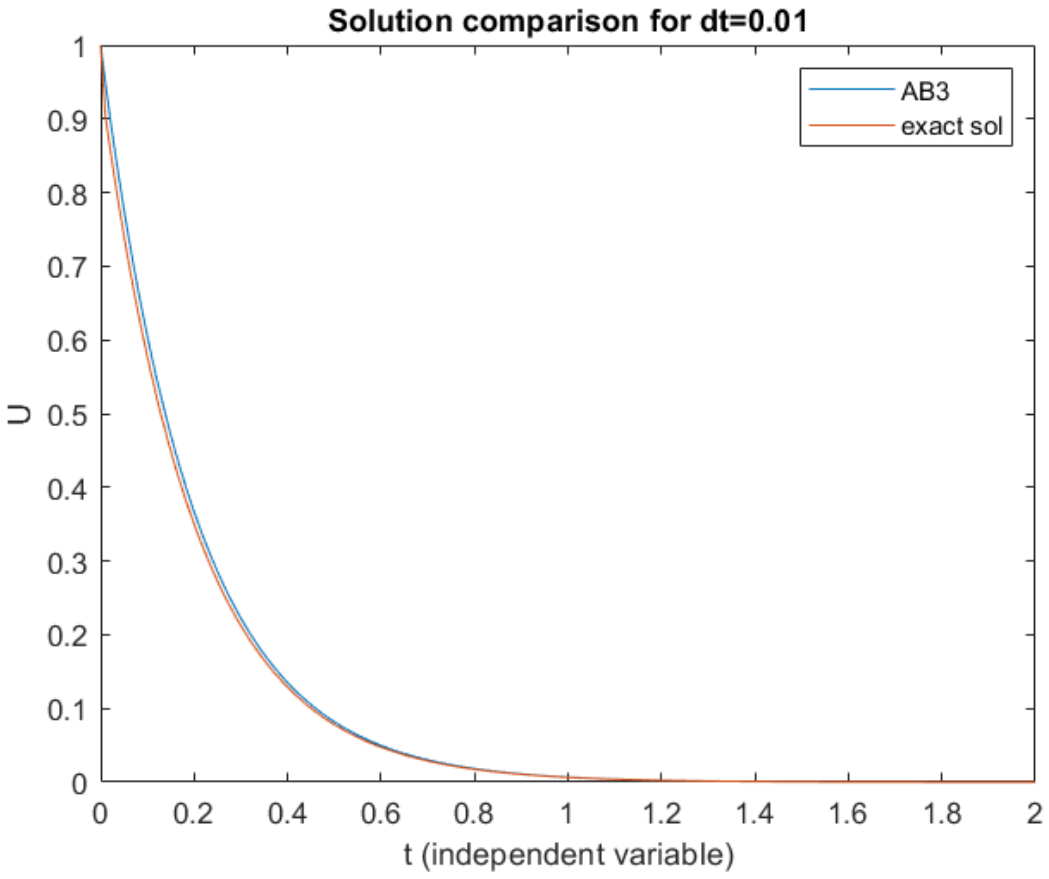
For timestep 0.15 seconds results show instability:




For 0.1 seconds, the solution converges:




For 0.01 seconds timestep, the curves almost coincide at the scale of the plot:

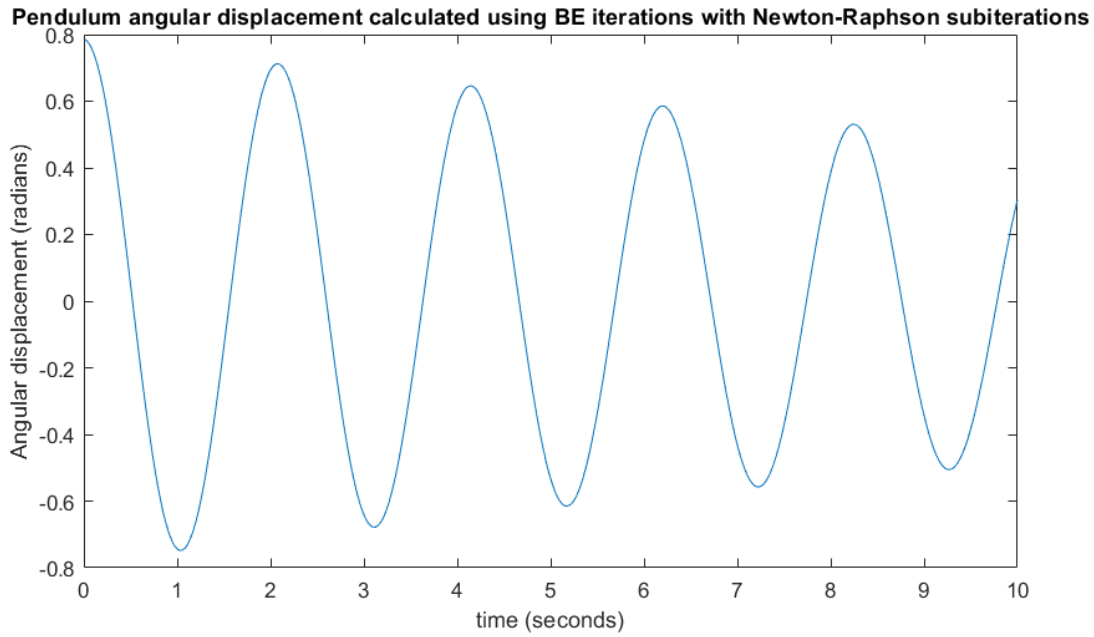


2B. The requested code for parts B and C (including implementations of Backward Euler and Trapezoidal using Newton Raphson subiterations):

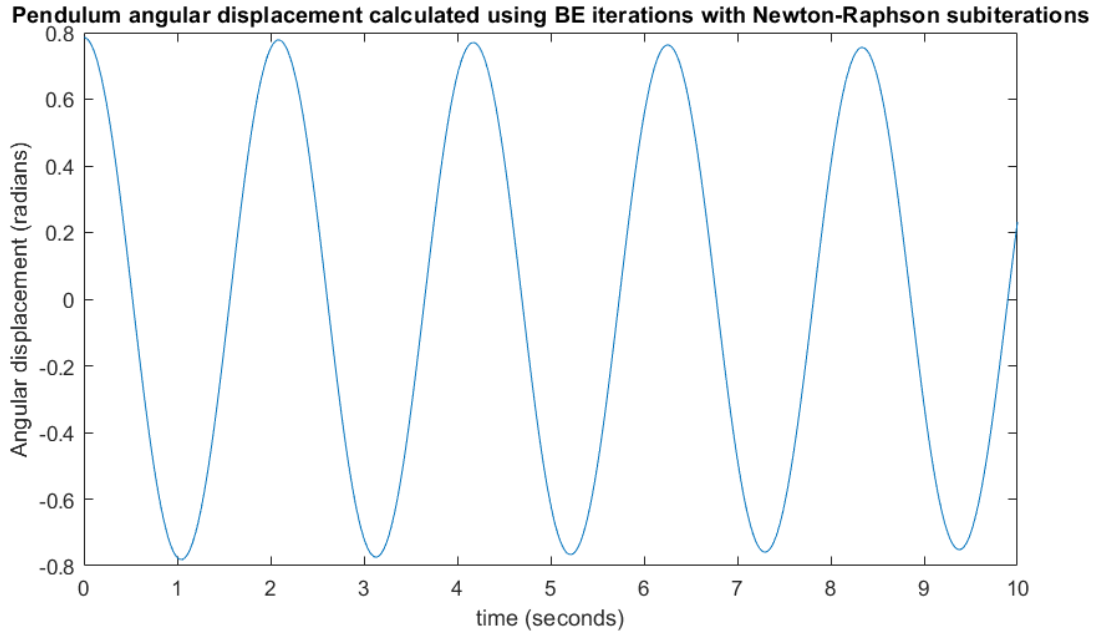
```
dt=0.01; %timestep
Tfinal=10;
t=[0:dt:Tfinal]; %time vector
u0=[0; pi/4]; %initial condition
%BE method
U=zeros(2,length(t));
U(:,1)=u0;
g=9.8;L=1; %gravitational acc and pendulum length
for i=2:length(t)
    w = U(:,i-1); %start with an initial guess of the new value
    normR=100;
while normR > 1e-12
    R = w - U(:,i-1) - dt*([-(g/L)*sin(w(2)); w(1)]); %Residual
    J = [0 -(g/L)*cos(w(2)); 1 0]; %Jacobian of the system
    dw = (eye(2)-dt*J)\-R; % solve for dw
    w = w + dw;
    normR = norm(R) ;
end
U(:,i)=w;
end
%trapezoidal integration method
U_trp=zeros(2,length(t));
U_trp(:,1)=u0;
g=9.8;L=1; %gravitational acc and pendulum length
for i=2:length(t)
    w = U_trp(:,i-1); %start with an initial guess of the new value
    normR=100;
while normR > 1e-12
    R = w - U_trp(:,i-1) - 0.5*dt*([-(g/L)*sin(w(2)); w(1)]+[-
(g/L)*sin(U_trp(2,i-1)); U_trp(1,i-1)]); %Residual
    J = [0 -(g/L)*cos(w(2)); 1 0]; %Jacobian of the system
    dw = (eye(2)-0.5*dt*J)\-R; % solve for dw
    w = w + dw;
    normR = norm(R) ;
end
U_trp(:,i)=w;
end

figure (1);
plot(t,U(2,:),'-',t,U_trp(2,:),'-')
legend('BE with NR','trapezoidal with NR');
title('Pendulum angular displacement calculated using BE/NR and
trapezoidal/NR');
xlabel('time (seconds)');
ylabel('Angular displacement (radians)');
```

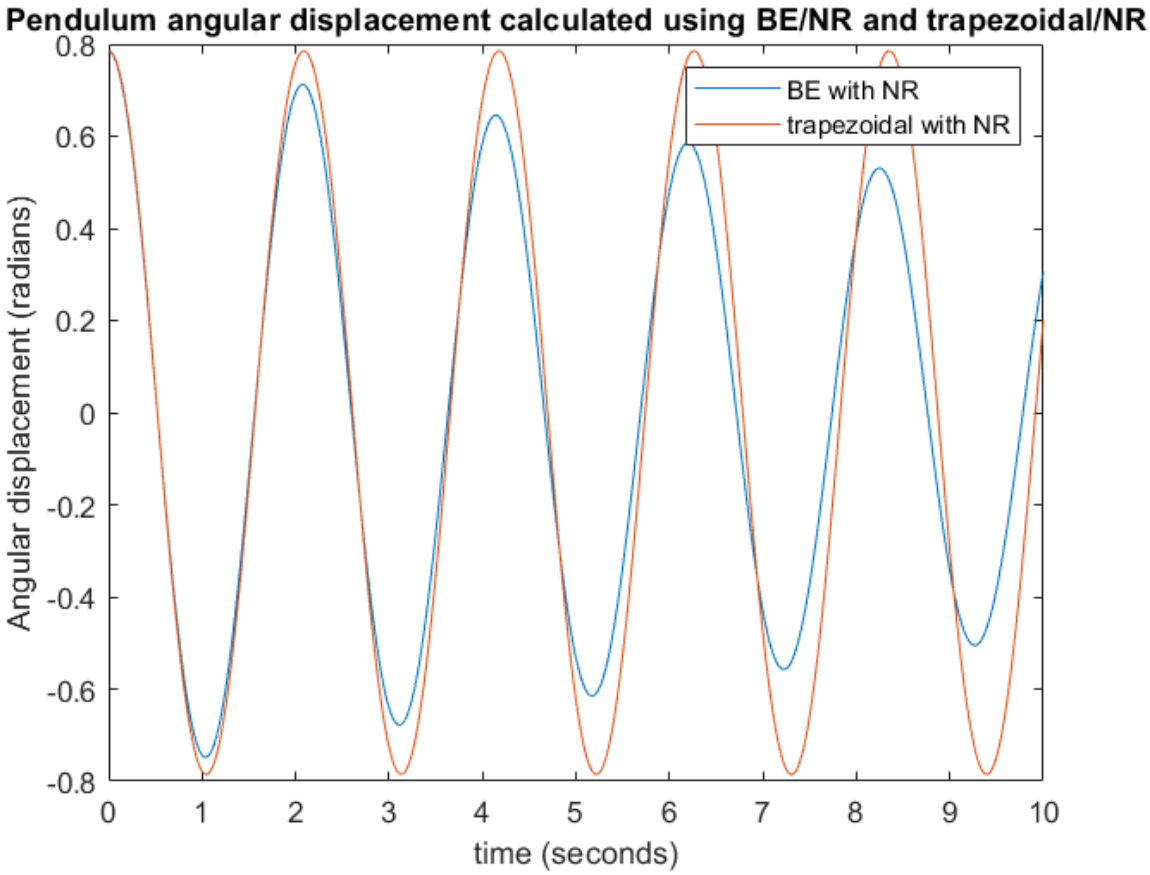
The resulting plot is made with dt=0.01 seconds:




Also for dt=0.001 seconds:




2C. Plot with dt=0.01 seconds:




2D. I wrote this code:

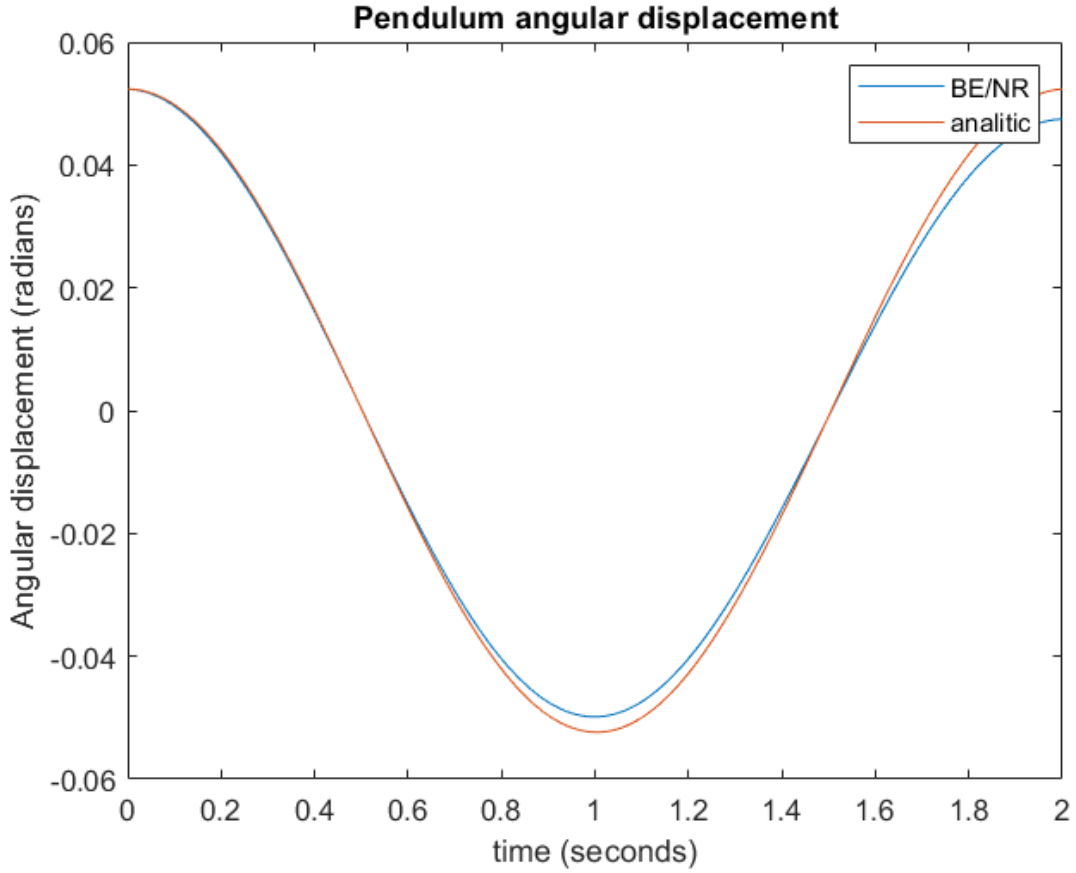
```
dt=0.01; %timestep
u0=[0; pi/4]; %initial condition
%BE method
g=9.8;L=1; %gravitational acc and pendulum length
omega=sqrt(g/L);
Tfinal=2*pi*sqrt(L/g);
t=[0:dt:Tfinal]; %time vector
U=zeros(2,length(t),59);
U_analitic=zeros(2,length(t),59);

for indice=1:59 %initialize 3d matrix wit angular amplitude, knowing
that initial ang speed is zero
    U(2,1,indice)=pi/(indice+1);
    U_analitic(2,1,indice)=pi/(indice+1);
end

for indice=1:59
    for i=2:length(t)
        w = U(:,i-1,indice); %start with an initial guess of the new
value
        normR=100;
    while normR > 1e-12
        R = w - U(:,i-1,indice) - dt*([-(g/L)*sin(w(2)); w(1)]);
%Residual
        J = [0 -(g/L)*cos(w(2)); 1 0]; %Jacobian of the system
        dw = (eye(2)-dt*J)\-R; % solve for dw
```

```
        w = w + dw;
        normR = norm(R) ;
    end
    U(:,i,indice)=w; %update matrix U at location i-1, indice with a
vector
    U_analitic(:,i,indice)=[-
U_analitic(2,1,indice)*omega*sin(omega*(i-
1)*dt),U_analitic(2,1,indice)*cos(omega*(i-1)*dt)];
    end
end
relative_errors=zeros(1,59);
for indice=1:59 %look for the maximal absolute relative error
    %relative_errors(indice)=norm(U_analitic(:,length(t),indice)-
U(:,length(t),indice))/norm(U_analitic(:,length(t),indice));
    relative_errors(indice)=norm(U_analitic(2,:,indice)-
U(2,:,indice))/norm(U_analitic(2,:,indice));
end
disp(relative_errors)
figure (1);
plot(t,U(2,:,59),'-',t,U_analitic(2,:,59),'-')
legend('BE/NR','analitic');
title('Pendulum angular displacement');
xlabel('time (seconds)');
ylabel('Angular displacement (radians)');
```

I have read values for relative errors displayed with the last disp() call. All initializations with angular amplitude less or equal 30 degrees give errors of less than 10% (on amplitude; I have seen the errors being larger for angular speed).
For pi/60 amplitude with dt=0.01 s I got:


Pendulum angular displacement
BE/NR
analitic
Angular displacement (radians)
time (seconds)
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2